Experiment #4: Continuous Process Control

Continuous process control involves maintaining desired process conditions. Heating or cooling objects to a certain temperature, holding a constant pressure in a steam pipe, or setting a flow rate of material into a vat in order maintain a constant liquid level, are examples of continuous process control. The condition we desire to control is termed the "process variable." Temperature, pressure, flow rate, and liquid level are the process variables in these examples. Industrial output devices are the control elements. Motors, valves, heaters, pumps, and solenoids are examples of devices used to control the energy determining the outcome of the processes.

The control action taken is based on the dynamic relationship between the output device's setting and its effect on the process. Generally speaking, process control can be classified into two types: open-loop and closed-loop. Closed-loop control involves determining the output device's setting based on measurement and evaluation during the process. In open-loop control, no automatic check is made to see whether corrective action is necessary.

A simple example of open-loop control would be cooling your bedroom on a hot summer evening. Your choices are using a window fan or an air conditioner. The window fan is a device that you set – low, medium, or high speed – based on your evaluation of what the situation needs for control. This evaluation involves an understanding of what the cause-and-effect relationship is of your speed setting vs. the room conditions. There is also an element of prediction involved. Once you make the setting decision, you are in for the night. You are setting up an open-loop control system. If your evaluations are correct, you will have a great night's sleep. If they are not, you may wake up shivering and cold or sweaty and hot! On the other hand, a room air conditioner allows you to set a certain desired temperature. A thermostat continuously compares the desired temperature with a measurement of actual room temperature. When room temperature is over the desired setpoint, the air conditioner is turned on. As the room cools below the setpoint, the air conditioner is turned off. As the night goes on and the outside temperature cools down, this closed-loop system will automatically spend less time on than off. This is an example of closed-loop feedback control, because the action is taken based on measurement of room temperature.

Which is better? Arguably, some people prefer air conditioning to a fan, but others do not. If the objective is to maintain a comfortable sleeping temperature, they both have their advantages. In terms of industrial control, the lower cost and simplicity of setting the window fan in an open-loop mode is very attractive. On the other hand, the automatic control of the closed-loop air conditioner ensures a more consistent bedroom temperature as the outside temperature changes.

Determining the best control action for an application and designing the system to provide this action is what the field of process control engineering is all about.

Microcontrollers have proven to be a dependable, cost-effective means of adding a level of sophistication to the simplest of control schemes. The next three exercises will focus on the characteristics of various methods of continuous control. We will develop an environment in which we can model process control, get process variable data into the BASIC Stamp, and study open-loop control principles. The first two items will take a little time and effort, but will be worthwhile, because the setup and circuitry will be used again for Experiments #5 and #6.

Temperature is by far the most common process variable that you will encounter. From controlling the temperature of molten metal in a foundry to controlling liquid nitrogen in a cryogenics lab, the measurement, evaluation, and control of temperature are critical to industry. The objective of this exercise is to show principles of microcontroller-based process control and enlighten you about interfacing the controller to real-world I/O devices. The exercises are restricted to circuits that fit on the Board of Education and to output devices that can be driven by its 9-volt, 300-mA power supply. As you monitor and control the temperature of a small environment, realize that through proper signal conditioning, the applications for which you can apply the BASIC Stamp are limitless.

Exercises

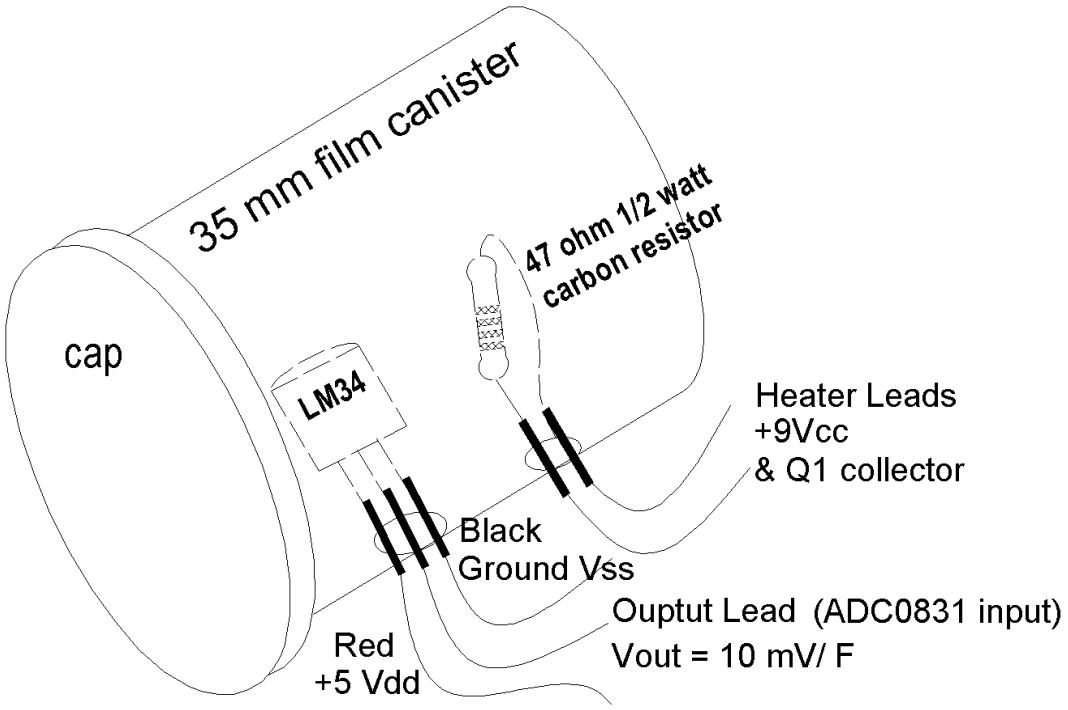
Exercise #1: Closed-Loop, On-Off Control

To set up our small environment, you will need the following parts:

- 35 mm plastic film container
- Six feet of 22 to 28 gauge hook-up wire
- 47-ohm, half-watt carbon resistor
- LM34DZ integrated circuit temperature sensor
- Electrical tape or heat-shrink tubing
- Soldering iron, solder, and wire cutter/strippers

You will put the 47-ohm resistor and the temperature sensor inside the 35-mm film canister. Leads from these devices will come back to the Board of Education. Placing the cap on the canister creates a closed environment. High current through the resistor will heat the environment, and the sensor will convert temperature to an analog voltage. A current-boost transistor from Experiment #3 will drive the resistor/heater, and you will add an analog-to-digital converter to get binary temperature information into the BASIC Stamp. Figure 4.1 depicts this construction step. Follow the procedures on the next page to construct the canister environment and signal-conditioning circuitry.

Figure 4.1: Film Canister Heated Environment



Preliminary Preparation

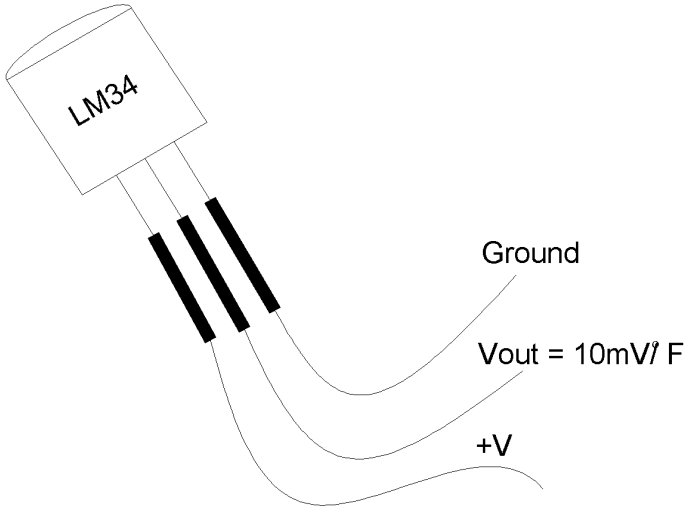
The 35-mm film canister will need two holes punched or drilled in it. A one-hole handheld paper punch could be used to create the holes. Place one hole as far toward the bottom as possible and the other near the top. The second hole should not be so near the top that it interferes with the cap. The sensor and the heater will be placed into these holes; some separation between the two is important.

In the last exercise, we ended with the ability to control the on-off status of a 47-ohm resistor acting as a heating element. The current-boost transistor acted as a switch, controlling the unregulated 9-volt supply to the resistor. As you should have seen, when the transistor turned on, the 9-volt supply was placed across the resistor and it became quite warm. It is no surprise considering that the power consumed is $P = V^2/R = 9^2/47$, or 1.7 watts! This is beyond the resistor's half-watt rating, but we are using it as a heater. It may become discolored, but should be all right otherwise.

Solder 12" lead wires onto the resistor so it can be placed in the film canister. A small amount of electrical tape or heat-shrink tubing over the connections should be used to avoid shorting in the canister. Place the resistor through the lower hole in the canister. Bend the leads and tape them down to the outside of the canister so the resistor is suspended in the middle of the canister.

Next, connect leads to the LM34 temperature sensor to act as a temperature probe. The LM34 precision IC temperature sensor is an excellent device in terms of its linearity, cost, and simplicity. Appendix D contains the datasheet for this very useful device. The sensor's output voltage changes 10 mV per degree Fahrenheit and is referenced at 0 degrees. With a DC power supply and a voltmeter, you have a ready-made Fahrenheit temperature sensor. Soldering some short leads onto the sensor and insulating them will result in a convenient probe. Twelve-inch lengths of red, black, and one other color wire will allow for easy identification of the +V, Ground, and Output leads. Short pieces of heat-shrink tubing or electrical tape should be used to insulate the leads from one another. Refer to Figure 4.2 and the device's datasheet in Appendix D.

Figure 4.2: LM34 Temperature Probe



Once your leads have been connected, test your temperature probe. Connect your probe to the +5-volt Vdd supply and ground. Use your voltmeter to monitor the LM34 output voltage of .01 volts per degree F. Simply move the decimal of the meter reading two places to the right to convert to temperature. For example; .75 V = 75 ^{o}F; .825 V = 82.5 ^{o}F; 1.05 V = 105 ^{o}F, etc. Then, try this:

- Complete your probe construction, then measure and record the room temperature.
- Hold the device between your fingers and watch the temperature rise.
- Hold it until the temperature becomes stable. How hot are your fingertips?
- The LM34 can measure temperatures up to 300 degrees. Briefly wave a flame under it and monitor higher temperatures.

Now, insert the sensor through the top hole of the film canister. Bend and tape its leads to the canister so the sensor is suspended inside. Cap your canister, and your model environment is complete. Keep in mind that although our laboratory setup is small and low power, it could represent controlling the temperature of a large kiln, a brewing vat, or an HVAC system. Appropriate output signal conditioning identified in Experiment #3 can allow the BASIC Stamp to control almost any industrial device.

Next, let's turn our attention to the Board of Education and set up the circuitry necessary for the experiment. Figure 4.3 represents the four circuits used in the exercises. All four circuits can fit on the Board of Education; you just have to be efficient with your use of the space. Figure 4.3a is a simple active-high pushbutton switch. It will be used to toggle the heater on and off. The output drive circuit depicted in Figure 4.3b also is very simple. Your board will contain the current-boost transistor (from Experiment 3) to drive the heater. Notice that an LED and current-limiting resistor have been added to indicate the status of this drive circuit. Note also that this circuit goes to the +5- Vdd supply, not the 9-volt unregulated supply.

Since the BASIC Stamp microcontroller does not have analog input capability, we must add an analog-to-digital converter to change the analog output of the LM34 temperature sensor to digital data. A one-step solution to signal conditioning is to use the serial analog-to-digital converter shown in Figure 4.3c. National's ADC0831 is a suitable A/D converter for our application, and will prove to be a very useful device for this and future applications. It's worth a moment at this point to look at the device's features and limitations.

This converter will take an input voltage and convert it to 8-bit digital data with a range of 256 possible binary representations. Potentiometers can be used to externally set the analog voltage range spanned by the 0 to 255_{10} digital output capability of the ADC0831. The voltage at Vin (-) Pin 3 sets the zero value. The Vref voltage at Pin 5 sets the voltage span above Vin (-) over which the resolution of 255 is spread.

Figure 4.3: Process Control Circuitry

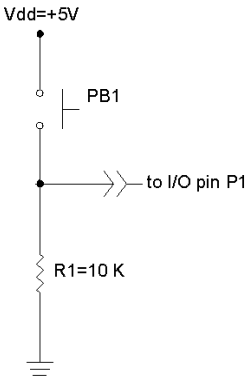


Figure 4.3a: Pushbutton

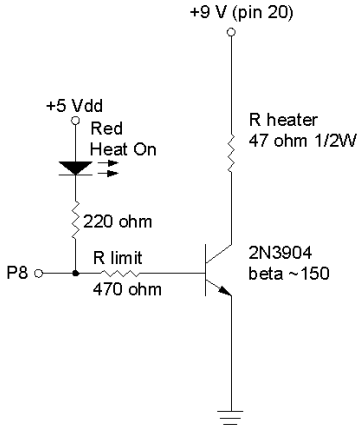


Figure 4.3b: BASIC Stamp Driving Heater with LED Indicator

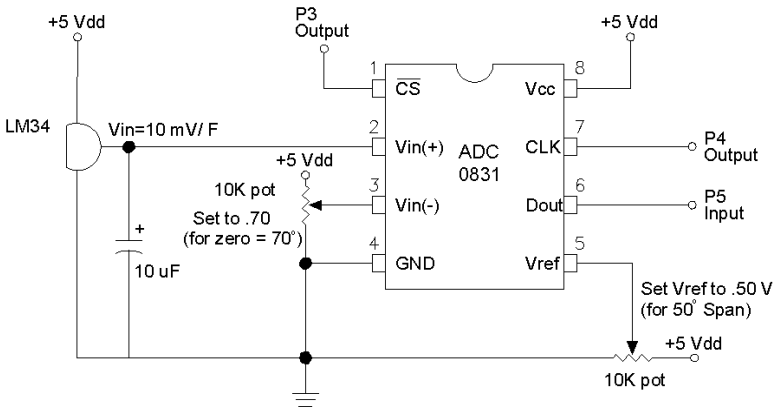


Figure 4.3c: ADC0831 Serial A-to-D

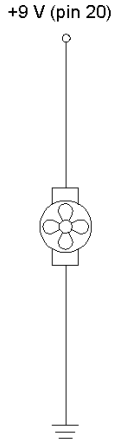


Figure 4.3d: Brushless Fan Directly Connected

These voltages are used to focus the range of the A/D device to cover the analog voltage being converted. The application will operate from room temperature (70^{o}) up to 120 ^{o}F. This temperature range equates to an LM34 voltage output of .70V to 1.20V. To maximize resolution, the span of interest is .5 V (1.2V-.7v); and the zero reference, termed the *offset*, is .7 V. Since we focus on just this range, each binary step represents approximately 0.2 degrees. This allows us to accurately resolve the temperature.

Construct the A/D converter circuit on the Board of Education. By using multi-turn trim potentiometers, the reference and span voltage levels can be set very accurately. Carefully measure and adjust these potentials. Attach the output of the LM34 to the input of the A/D converter.

Controlling the ADC0831 is relatively simple. A program control line tells the device to make a conversion. Once a conversion has been performed, the binary value can be output one bit at a time to the BASIC Stamp. The serial flow of data from the converter is controlled by output pins of the BASIC Stamp driving the "chip select" and "clock" lines. The chip select line (CS) is set low, followed by a low-to-high clock pulse. This starts the conversion. Subsequent clock pulses initiate the transfer of each binary bit starting with the most significant bit first. Parallax provides a convenient instruction, called **`shiftin`**, specifically designed for controlling synchronous serial communication. Once the binary data is clocked into the BASIC Stamp, it is converted to temperature, based on the zero and spanning values. (The use of the ADC0831 is detailed in the Parallax Basic Analog and Digital text, which can be used as a further reference to this section.)

Program 4.1 has been written to test your converter and driver circuits, and exercise the StampPlot Lite Interface. The first section of the program configures StampPlot Lite. A section that establishes variables and constants follows this. The program is designed for the circuit of Figure 4.3. Double-check the proper connections to I/O Pins 1,3,4,5 and 8. Accurately set the Zero and Span voltages of the ADC0831 to .7 and .5, respectively.

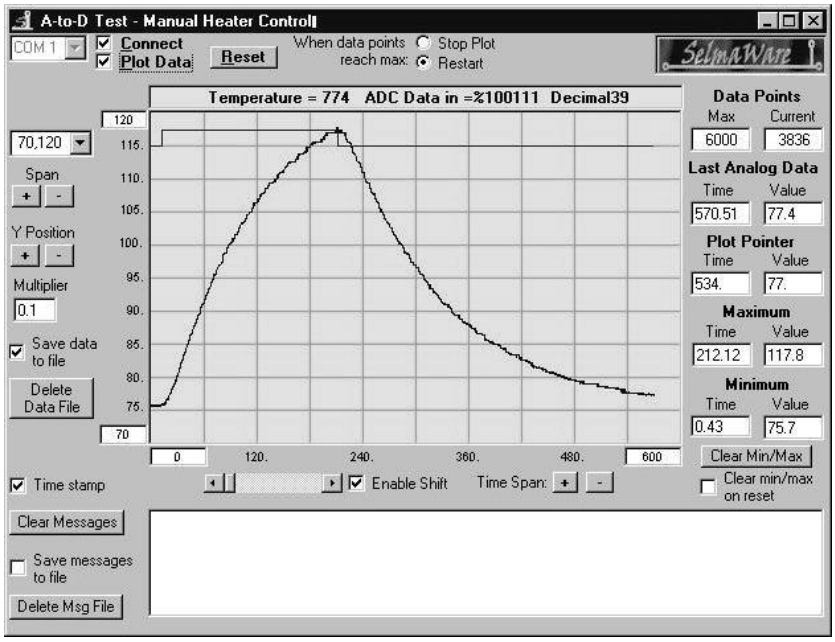
Load the program. Running the program will result in the DEBUG window opening and scrolling values and messages to the screen. Close the DEBUG window and open the StampPlot Lite Interface. At this point, select the appropriate COM port and check "Connect." Momentarily press the "Reset" button on the BASIC Stamp Board of Education to load the configuration and begin plotting the data. The user-status box will report the current temperature and the output of the A/D converter's binary and decimal values. The temperature and status of the heater will be plotted on the interface. Pressing PB1 will **`toggle`** the heater ON and OFF. (Feel free to omit the comments from this code, if you wish).

```
'Program 4.1: Analog-to-Digital & ON-OFF test with StampPlot Interface

'Pushbutton P1 toggles the heater fully ON and OFF.  It then establishes
'constants and variables used to acquire data from the ADC0831 serial A-to-D.
'StampPlot is used to graphically display results.  Program assumes that the
'circuitry is set according to Figure 4.3. ‘ADC0831: "chip select" CS = P3, "clock"
'Clk=P4, & serial 'data output"Dout=P5.  ‘Zero and Span pins: Digital 0 = Vin(-) =
'.70V and Span = Vref = .50V.


'Configure Plot
Pause 500                          'Allow buffer to clear
DEBUG "!RSET",CR                   'Reset plot to clear data
DEBUG "!TITL HEATER CONTROL SAMPLE",CR  'Caption form
DEBUG "!PNTS 6000",CR              '2000 sample data points
DEBUG "!TMAX 600",CR               'Max 300 seconds
DEBUG "!SPAN 70,120",CR            '50-300 degrees
DEBUG "!AMUL .1",CR                'Multiply data by .1
DEBUG "!DELD",CR                   'Delete Data File
DEBUG "!SAVD ON",CR                'Save Data
DEBUG "!TSMP ON",CR                'Time Stamp On
DEBUG "!CLMM",CR                   'Clear Min/Max
DEBUG "!CLRM",CR                   'Clear Messages
Debug "PLOT ON”,CR                 'Start Plotting
DEBUG "!RSET",CR                   'Reset plot to time 0

' Define constants & variables

CS con 3                           ' 0831 chip select active low from BS2 (P3)
CLK con 4                          ' Clock pulse from BS2 (P4) to 0831
Dout con 5                         ' Serial data output from 0831 to BS2 (P5)
Datain var byte                    ' Variable to hold incoming number (0 to 255)
Temp var word                      ' Hold the converted value representing temp

TempSpan var word                  ' Full Scale input span in tenths of degrees.
TempSpan = 5000                    ' Declare span.  Set Vref to .50V and
                                   ' 0 to 255 res. will be spread over 50
                                   ' (hundredths).

Offset var word                    ' Minimum temp.  @Offset, ADC = 0
Offset = 700
                                   'Declare zero Temp.  Set Vin(-) to .7 and
                                   'Offset will be 700 tenths degrees.  At these
                                   'settings, ADC output will be 0 - 255 for temps
```

```
                                'of 700 to 1200 tenths of degrees.

Wkspace1  var  byte             ' Workspace for the PB1's BUTTON command
Wkspace1 = 0                    ' Clear the workspace before using BUTTON
LOW 8                           ' Initialize heater OFF

Main:
  GOSUB Getdata
  GOSUB Calc_Temp
  GOSUB Control
  GOSUB Display
  GOTO Main

Getdata:                        'Acquire conversion from 0831
  LOW CS                        'Select the chip
  LOW CLK                       'Ready the clock line.
  PULSOUT CLK,10                'Send a 10 uS clock pulse to the 0831
  SHIFTIN Dout, CLK, msbpost,[Datain\8]       'Shift in data
  High CS                       'Stop conversion
RETURN

Calc_Temp:                      'Convert digital value to
  Temp = TempSpan/255 * Datain/10 + Offset     'temp based on Span &
RETURN                    'Offset variables.

Control:                  ' Manual heater control
  Button 1,1,255,0,Wkspace1,1,Toggle_it
RETURN

Toggle_it:
  TOGGLE 8
  RETURN

Display:                  'Plot Temp, binary ADC, & Temp status
  DEBUG DEC Temp,CR
  DEBUG IBIN OUT8,CR
  DEBUG "!USRS Temperature = ", DEC Temp,"   ADC Data in =%", BIN Datain,  "
Decimal", DEC Datain, CR
RETURN
```

The StampPlot Lite interface will give you a dynamic representation of temperature changes in your canister. Toggle the heater ON and OFF and watch the response. The screen shot in Figure 4.4 represents the closed canister heating to 120 degrees and then cooling after the heater is turned off. Play with your system to become more familiar with its response; then, let's take a little closer look at the subroutines that make up the program.

Figure 4.4: Screen Shot using Program 4.1

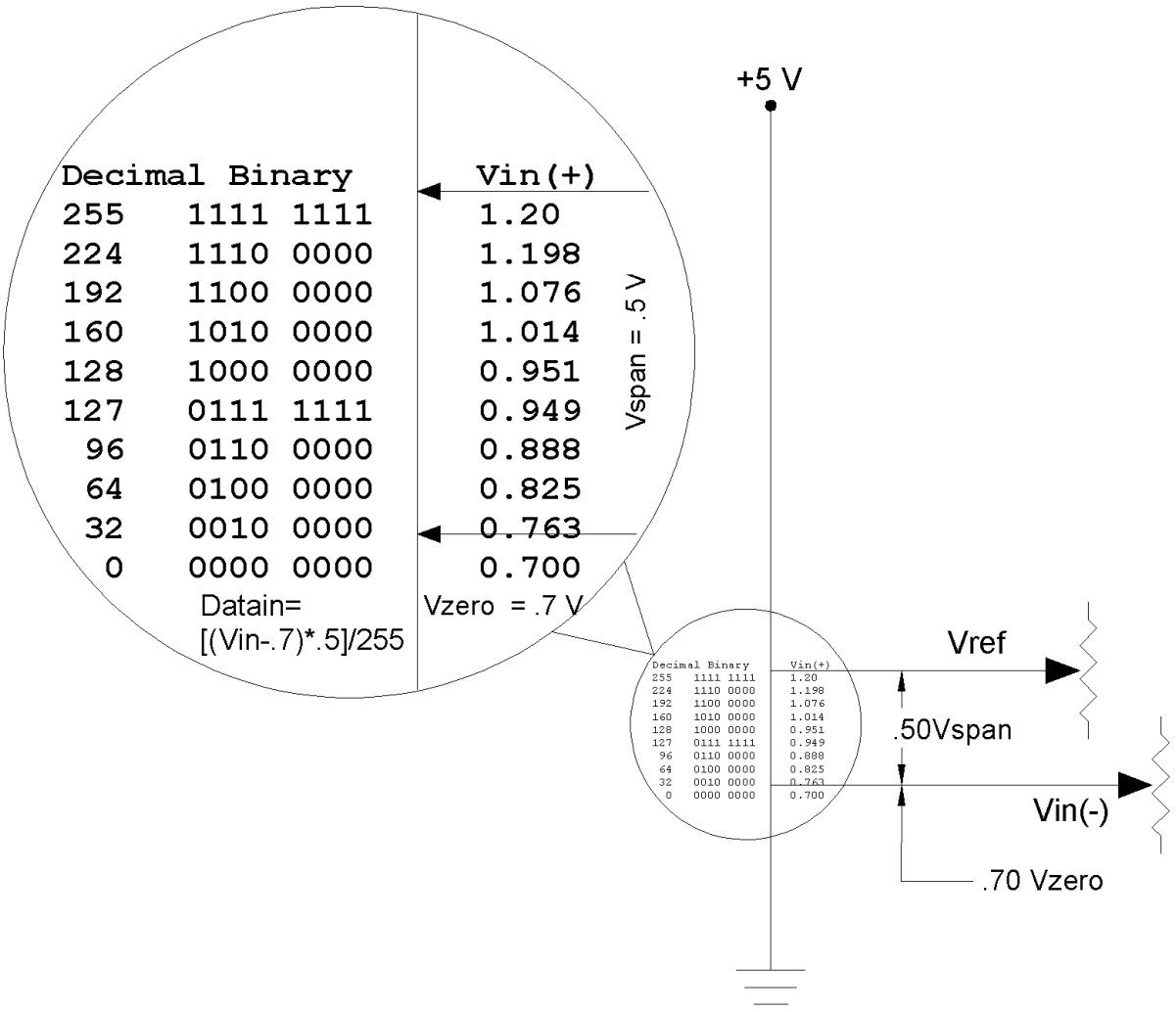


The main loop of this program simply executes three subroutines, **`Getdata`**, **`Calc_Temp`**, and **`Display`**. When running, the BASIC Stamp jumps back to the **`Getdata`** subroutine first. The last line of this routine instructs the processor to **`RETURN`** to the main loop and executes the next instruction, **`GOSUB Calc_Temp`**. The **`Calc_Temp`** subroutine executes, and it ends with a return. The BASIC Stamp returns to **`GOSUB_Display`**. After **`Display`** executes, its **`RETURN`** goes back to the instruction of **`GOTO Main`** and the process starts over. This is an organized approach to structuring our program. Later, when we include evaluation and control in our program, we simply add another subroutine, such as **`GOSUB_Control.`**

Let's take a closer look at the two primary subroutines of program 4.1. The **`Getdata`** subroutine begins with a high-to-low transition on the "chip select" line. This readies the A/D for operation.

The **LOW CLK** and **pulsout CLK,10** instructions tell the A/D converter to make a conversion of the Vin(+) voltage at this time. The ADC0831 is an 8-bit successive approximation converter. It's 256 possible digital combinations are spread over a voltage range determined by the potentials at the $V_{in(-)}$ and V_{ref} pins. $V_{in(-)}$ defines the voltage for which 0000 0000 would be the conversion. V_{ref} defines the range of input voltages above this point over which the other 255 digital combinations are spread. Figure 4.5 represents the Zero and Span settings for our application.

Figure 4.5: Zero and Span Settings for Our Application



With these settings, the ADC0831 is focused on a temperature range of 70 to 120 degrees. There can be an infinite number of possible temperature values within the .7 to 1.2-volt output range of the LM34. Only a few representative values are given. Since the 8-bit A/D converter has a resolution of 255, it can resolve this range of 50-degree temperatures to within .31 degrees. The conversion will be a binary number equal to [(Vin - .7) /.5] *255. Let's try a value within the range. Let's say the temperature is 98.6, which results in an LM34 output of .986 volts.

If Vin = .986, what would be the binary equivalent?

> [(.986-.7)/.5] *255 = 145.86. The answer is truncated to the whole integer of 145.
> The binary word would be 1001 0010.
> The binary conversion will be held and ready for transfer.

The **`shiftin`** instruction is designed for synchronous communication between the BASIC Stamp and serial devices such as the ADC0831. The syntax of the instruction is **`SHIFTIN dpin, cpin, mode, [result\bits]`**. The parameters indicate:

- which pin data will arrive on (dpin),
- which pin is the clock (cpin),
- which bit comes first, the least significant (LS) or most significant (MS), and on which edge of the clock it is released, rising (PRE) or falling (POST),
- and, what the word width is and where you want it stored [Datain\8].

For our system, we previously declared Pin 5 as dpin and Pin 4 as the clock (CLK) pin. The ADC0831 outputs the most significant bit first on the trailing edge of the clock. Therefore, MSPOST is the mode. And, finally, the 8-bit data will be held in a byte variable that we declared as **`Datain`**.

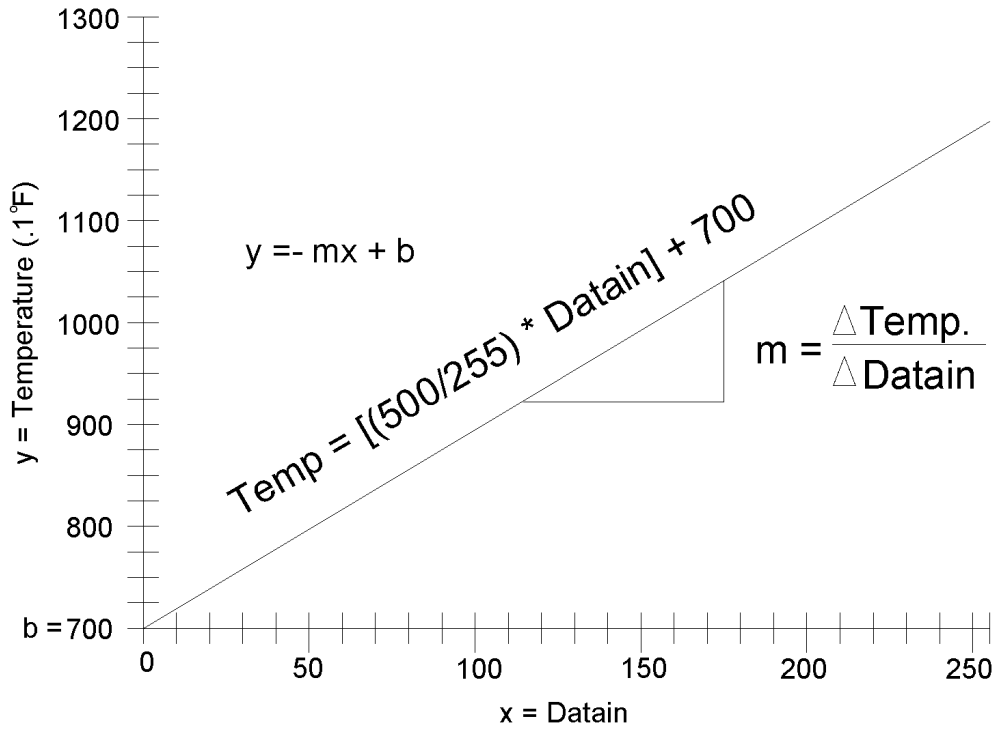
After the binary data is brought into the BASIC Stamp, it is available for our program to use. It would be most convenient to use if it were expressed in terms of the actual measurement units. For our application, that would be in degrees. The next subroutine, **`Calc_Temp`**, does just that. By knowing the zero and span transfer function of the conversion process, we use the standard y = mx + b formula. Where: y =Temperature, m = slope of the transfer function, and b is the offset. Temperature will be resolved and expressed in tenths of degrees.

Refer to the Calc_Temp formula: **`Temp = Tempspan/255 * 255/10 + 700`**

To increase the accuracy in resolving the slope (m), the **Tempspan** variable is scaled up by 10, to 5000 hundredth degrees. The slope is therefore, 5000/255 ~ 19 or .19 degrees per bit. Multiplying 19 times **Datain** tells you how far the measurement is into the span. This is in one one-hundredth of a degree at this point; therefore, divide by 10 to scale it back to tenths. Adding this to the Zero value of 700 (70 degrees) results in the actual temperature in tenths of a degree. Resolution is approximately .2 degrees over a range of inputs from 70.0 to 120.0 degrees.

The graph in Figure 4.6 plots the transfer function of the input A/D decimal equivalent input to temperature of the canister. Changing the span of coverage changes the slope of the transfer function. Changing the Zero value changes the y intercept.

Figure 4.6: Transfer Function



An additional word of caution about the BASIC Stamp math operation:

- A formula will be executed from left to right unless bracketing is used to set precedence.
- At no point can any subtotal exceed 32,759 or -32,760.
- Also, all remainders will be truncated, not rounded up.

Challenge: Change the Zero and Span voltages and edit the program to match the new range.

1. Your system should be able to raise the temperature of the closed canister beyond the 120-degree limit set by Program 4.1. Change the Zero and Span potentiometers for coverage of a temperature range from 75 degrees to 200 degrees. This allows for a wider range of coverage, but what is the resolution of your system now? Be patient, and let your system stabilize. Record the maximum temperature of your system.

2. Set the Zero and Span of your system to focus on the very narrow range of one degree below your room temperature to four degrees above it. Set the **`Calc_Temp`** variables to display in hundredths of degrees. Track these changes by leaving the cap off of the canister and simply touching the sensor with your warm finger. As you see, the resolution is great, but the trade-off is a decreased range of operation.

Having the ability to control the span and reference of the ADC0831 allows you to focus on a range of analog input. This helps maximize the resolution and accuracy of your system. The following exercise will require the original range of 70 to 120 degrees. Return the Zero and Span potentiometers back to .7 and .5 volts, respectively.

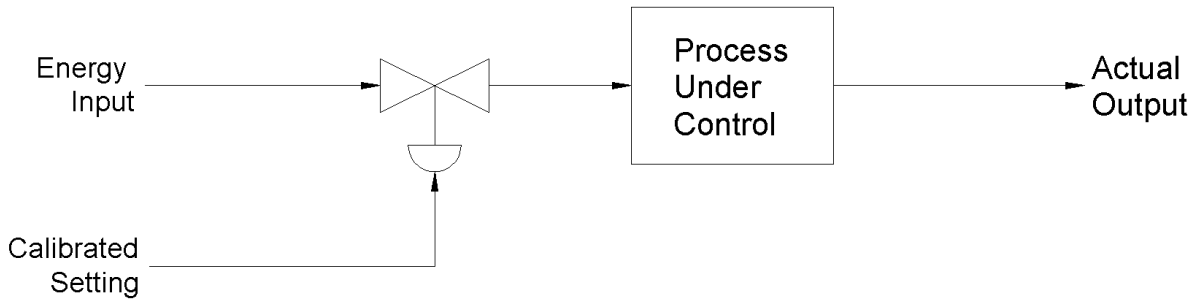
Now, after all of that, we can get back to a study of control theory!

Exercise #2: Open-Loop vs. Closed-Loop Control

Open-Loop Control

The simplest form of control is open loop. The block diagram in Figure 4.7 represents a basic open-loop system. Energy is applied to the process through an actuator. The calibrated setting on the actuator determines how much energy is applied. The process uses this energy to change its output. Changing the actuator's setting changes the energy level in the process and the resulting output. If all of the variables that may affect the outcome of the process are steady, the output of the process will be stable.

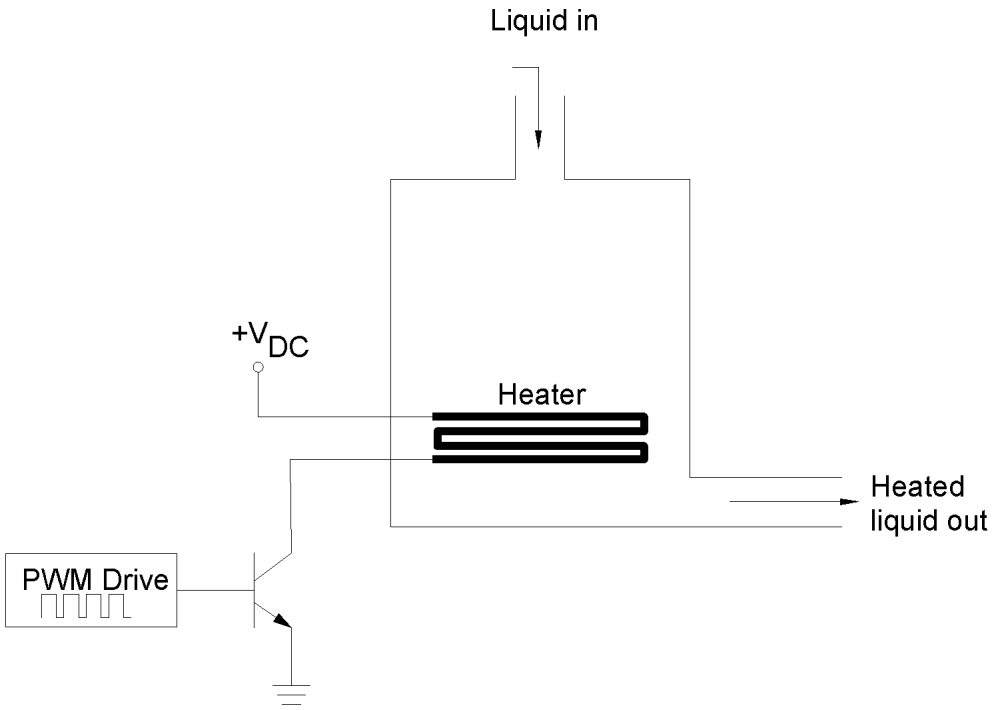
Figure 4.7: Open-Loop Control



The fundamental concept of open-loop control is that the actuator's setting is based on an understanding of the process. This understanding includes knowing the relationship of the effects of the energy on the process and an initial evaluation of any variables disturbing the process. Based on this understanding, the output "should" be correct. In contrast, closed-loop control incorporates an on-going evaluation (measurement) of the output, and actuator settings are based on this feedback information.

Consider the temperature control process shown in Figure 4.8. The material being drawn from the tank must be kept at a 101^{o} temperature. Obviously, this will require adding a certain amount of heat to the material. The drive on the transistor determines the power delivered to the heating element. The question becomes "How much drive is necessary?"

Figure 4.8: Open-Loop Heating Application



For a moment, consider the factors that would affect the output temperature. Obviously, ambient temperature is one. Can you list at least three others? How about:

- The rate at which material is flowing through the tank.
- The temperature of the material coming into the tank.
- And, the magnitude of air currents around the tank.

These are all factors that represent BTUs of heat energy taken away from the process. Therefore, they also represent BTUs that must be delivered to the process if the desired output is to be achieved. If the drive on the heating element were adjusted to deliver the exact BTUs being lost, the output would be stable.

In theory, the drive level could be set and the desired output would be maintained continuously, as long as the disturbances remained constant.

Let's now assume that it is your objective to keep the interior of your film canister at a constant temperature. A good real-world example would be that of an incubator used to hatch eggs. To hatch chicken eggs, it is important to maintain a 101°F environment.

Turning on the heater will warm up the interior of the canister. In our earlier test, you turned on the heater's drive transistor, and the temperature rose above 101°F. Obviously, to maintain the desired temperature, we will not need to have full power applied to the resistor. Through a little testing, you can determine just what drive level would be needed to yield the correct temperature.

The drive to the power transistor in Figure 4.8 is labeled as PWM. This is the acronym for pulse-width modulation. PWM is a very efficient method of controlling the average power to loads such as heating elements. The square wave is driving the transistor as a current-sinking switch. When the drive is high, the transistor is saturated, and full power is applied to the heater. A logic Low applied as base drive puts the transistor in cutoff; therefore, no current is applied to the load. Multiplying the percentage of the total time that the load receives full power times the full power will give the average power to the load. This average on-time is the duty cycle and is usually stated as a percentage. A 50% duty cycle would equate to half of the full power drive, 75% duty cycle is three-quarters full power, etc. It was stated earlier that the 47-ohm "heater" resistor in our canister would receive 1.7 watts when fully powered by the 9-volt unregulated source supply. The pushbutton switch was used to toggle the power on and off. If you were to press the switch rapidly at a constant rate, the resistor would receive 1.7 watts during the ON time and 0 watts during the OFF time. This 50% duty cycle would result in an average power consumption of .85 watts ($P_{average} = P_{full} * \text{duty cycle}$). Complete the table in Figure 4.9 below for power consumed at duty cycles of 75% and 25% for your system.

Figure 4.9: Average Power

$P_{average} = P_{full} * \text{duty cycle}$		
Full Power (P_{full})	Duty cycle	Average Power (P_{avg})
1.7 W	100%	1.7 W
1.7 W	75%	
1.7 W	50%	0.85 W
1.7 W	25%	
1.7 W	0%	0 W

PBASIC provides a useful instruction for providing pulse-width modulation.
Its syntax is: **`PWM pin, duty, duration`**

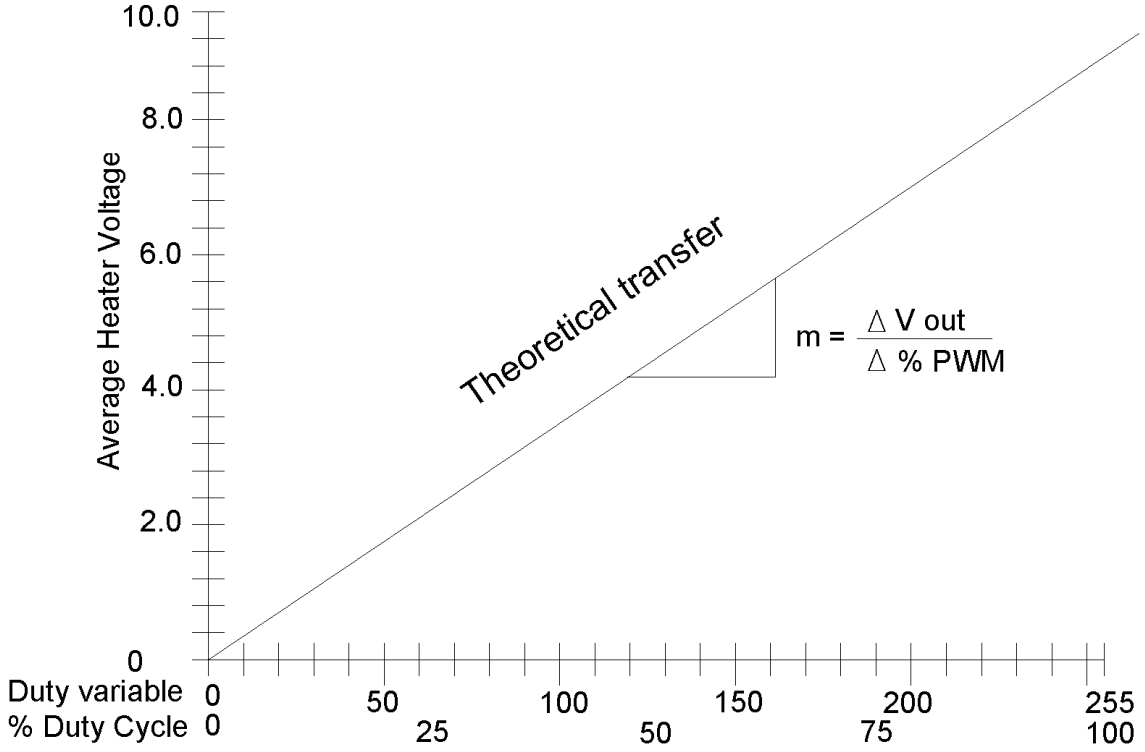
Where: **`pin`** is the output pin you are driving.
`duty` is the duty cycle relative to 255 being 100%.
`duration` is the window of time in milliseconds over which the duty cycle is provided.

Challenge! Graphing PWM duty vs. Vout

Use your multi-meter to measure the average voltage across the heating element at various PWM commands. Change the duty variable in Program 4.2 to increments between 0 and 255. Plot the average voltage on the graph in Figure 4.10.

```
'Program 4.2:  PWM vs. Vout

' Change the Duty = 50 in increments of 10 between 0 and 100.  Measure the
' average output voltage that results.

DutyCycle    var    byte
Duty         var    byte

DutyCycle = 50                          ' Begin with a 50% duty cycle

Loop:
  Duty = (DutyCycle * 255/100)          'Scale DutyCycle to PWM (0-255) duty
  PWM 8, Duty, 200                      ' Apply a Duty Cycle of to the heater
  DEBUG " Testing at a Duty Cycle of  ",  DEC DutyCycle, "%.", CR
GOTO Loop
```

Figure 4.10: Graph of Heater Voltage vs. PWM Duty Cycle



If you are not familiar with PBASIC's PWM instruction, refer to the BASIC Stamp Manual Version 1.9 pp. 293-295. One aspect of using the command should be understood. PWM applies pulses for a period of time defined by the duration value. During the time when the rest of the program is executing, there is no output applied to the load. As a result, the average voltage at a 100% duty cycle (duty =255) will result in a value less than the full voltage expected. The slower your program cycle time, the greater is this disparity. To get a better understanding of cycle time, place a **PAUSE 200** in Program 4.2. Compare the resulting output voltage with earlier readings. Change the length of the pause and notice the results.

Challenge! Analyzing your Open-Loop System

The following program is developed to study the relationship between PWM drive on your heater and the resulting stable temperature. The program will apply PWM drive levels in 10% increments. Each increment will last approximately four minutes. The program will end after 100% drive has been applied. StampPlot Lite will give you a graphical representation of your system's response, along with time stamp information in the list box. Furthermore, if you are really interested, the StampPlot Lite data file can be imported into a spreadsheet, applied to a graph and analyzed.

Figure 4.11: Screen Shot of PWM Drive vs. Temperature

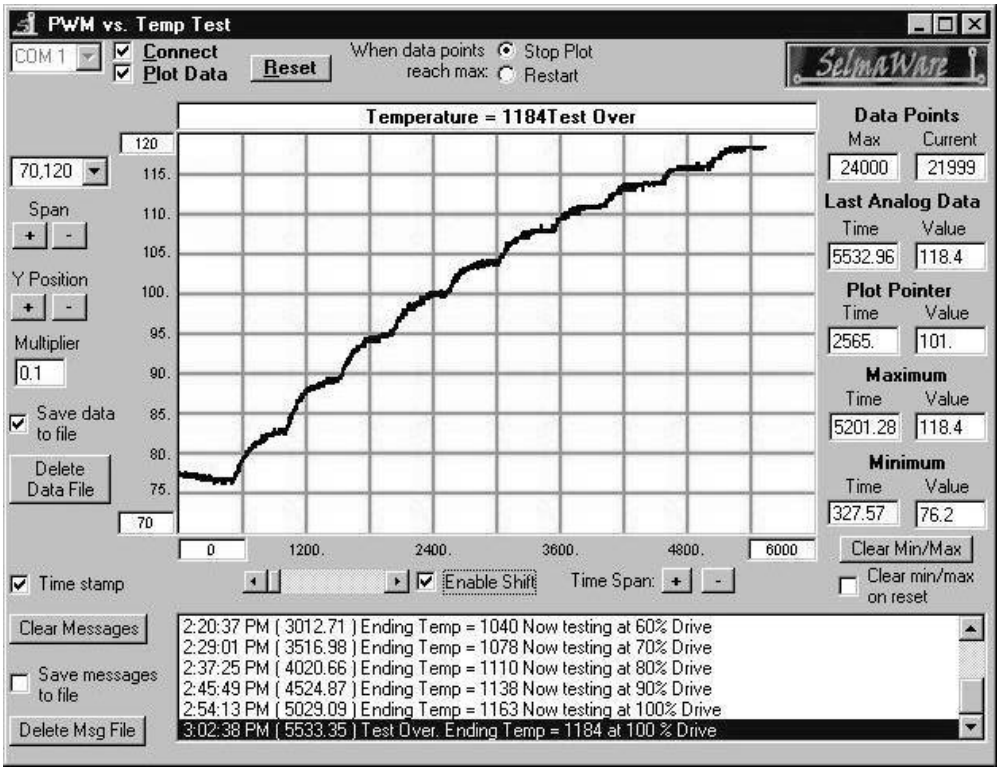


Figure 4.11 is typical of a StampPlot Lite screen shot resulting from this test. Load Program 4.3. Before running the program, be sure your canister has cooled to room temperature. Place the cap on your canister and start the program. When the DEBUG window appears, close it and start StampPlot Lite. Connect using StampPlot Lite and press the restart button to reload the program and begin the test.


```
'Program 4.3: PWM vs. Temp Test with StampPlot Interface

'This program tests the canister's temperature rise for incremental increases of
'PWM drive. Program runtime is approximately 40 minutes.  This can be adjusted
'by 'changing the "tick" and/or "Drive" increments.
'Program assumes that the circuitry is set according to Figure 4.3.
'ADC0831: '"chip select" CS = P3, "clock" 'Clk=P4, & serial data output"Dout=P5.
'Zero and Span pins: Digital 0 = Vin(-) = .70V and Span = Vref = .50V.

'Configure Plot
Pause 500                          'Allow buffer to clear
DEBUG "!RSET",CR                   'Reset plot to clear data
DEBUG "!TITL PWM vs. Temp Test",CR 'Caption form
DEBUG "!PNTS 24000",CR             '2000 sample data points
DEBUG "!TMAX 6000",CR              'Max 300 seconds
DEBUG "!SPAN 70,120",CR            '50-300 degrees
DEBUG "!AMUL .1",CR                'Multiply data by .1
DEBUG "!DELD",CR                   'Delete Data File
DEBUG "!SAVD ON",CR                'Save Data
DEBUG "!TSMP ON",CR                'Time Stamp On
DEBUG "!CLMM",CR                   'Clear Min/Max
DEBUG "!CLRM",CR                   'Clear Messages
DEBUG "!PLOT ON",CR                'Start Plotting
DEBUG "!RSET",CR                   'Reset plot to time 0

' Define constants & variables

CS con 3                           ' 0831 chip select active low from BS2 (P3)
CLK con 4                          ' Clock pulse from BS2 (P4) to 0831
Dout con 5                         ' Serial data output from 0831 to BS2 (P5)
Datain var byte                    ' Variable to hold incoming number (0 to 255)
Temp var word                      ' Hold the converted value representing temp

TempSpan var word                  ' Full Scale input span in tenths of degrees.
TempSpan = 5000                    ' Declare span.  Set Vref to .50V and
                                   ' 0-255 res. will be spread over 50
                                   '(hundredths).

Offset var word                    ' Minimum temp.  @Offset, ADC = 0
Offset = 700                       ' Declare zero Temp.  Set Vin(-) to .7 and
```

```
                                ' Offset will be 700 tenths degrees.  At these
                                ' settings, ADC output will be 0 - 255 for temps
                                ' of 700 to 1200 tenths of degrees.

LOW 8                           ' Initialize heater OFF

Drive var word                  ' % Drive
duty var word                   ' variable for PWM duty cycle
tick var word



Drive = 0                       ' Initialize variable to 0
tick = 0
duty = 0

' Get and display initial starting values.

  GOSUB Getdata
  GOSUB Calc_Temp
  DEBUG "Temp = ", DEC Temp, " Duty = ", DEC duty,CR
  DEBUG "!USRS Begining Test! -- Testing at ", DEC Drive, "% Drive.",CR

Main:                           ' main loop
  PAUSE 10
  GOSUB Getdata
  GOSUB Calc_Temp
  GOSUB Control
  GOSUB Display
  GOTO Main

Getdata:                        'Acquire conversion from 0831
  LOW CS                        'Select the chip
  LOW CLK                       'Ready the clock line.
  PULSOUT CLK,10                'Send a 10 uS clock pulse to the 0831
  SHIFTIN Dout, CLK, msbpost,[Datain\8]       'Shift in data
  High CS                       'Stop conversion
RETURN

Calc_Temp:                      'Convert digital value to
  Temp = TempSpan/255 * Datain/10 + Offset 'temp based on Span &
RETURN                          'Offset variables.

Display:                        'Plot present temperature
  DEBUG DEC Temp,CR
RETURN
```

```
Control:                          ' Testing system at different % duty cycles
  PWM 8,duty,200                  ' PWM
  tick = tick + 1                 ' increment tick variable
  IF tick = 2000 Then Increase    ' Program cycles per drive level change
RETURN

Increase:                         'Bump up the drive
  Drive = Drive + 10              'Drive increments = 10%
  duty = (Drive * 255/100)        'Scale %Drive to Duty
  If duty > 256 Then Stopit       'Stop test after 100% PWM
  DEBUG "Ending Temp = ", DEC Temp, " Now testing at ", DEC Drive, "% Drive", CR
  DEBUG "!USRS Testing at ", DEC Drive, "% Drive",CR
  tick = 0
RETURN

Stopit:                           ' Stop and print summary
  DEBUG "Test Over. Ending Temp = ", DEC Temp," at 100 % Drive",CR
  DEBUG "!USRS Temperature = ", DEC Temp,"Test Over",CR
END
```

Challenge! Open-Loop Control -- Desired Setpoint = 101° Fahrenheit

It is our objective to maintain a constant canister temperature of 101° Fahrenheit. Follow these procedures. Record values in the table of Figure 4.12.

1. Study the StampPlot Lite analysis that resulted from running Program 4.2. From the Text Box listing, record in the table the beginning ambient temperature, the temperature at the end of the 50% drive test, and the ending maximum temperature after 100% drive.

2. Use your cursor to find the Drive level that resulted in a temperature of 101 degrees.

3. Next, modify the **`Control`** subroutine of Program 4.2 so that the duty cycle remains at the constant value declared initially. Do this by removing the two lines indicated below.

```
Control:                                ' Testing system at different % duty cycles
       PWM 8,duty,200                   ' PWM
       tick = tick + 1                  ' increment tick variable
       IF tick = 2000 Then Increase     ' Program cycles per drive level change
       RETURN
```

4. At the beginning of the program, declare the **`DutyCycle`** to be the value that yielded 101° in our test StampPlot Lite. Run the program and allow the system to stabilize. How close was your estimation? Bump it up or down accordingly to find the setting that yields the desired result. In line #5 of the table, record the percent of drive that places the system at or near 101 degrees. Let it run for a moment and take note of the system stability. Once a drive setting has been established, an open-loop system will stabilize; and, as long as the disturbances that affect the process stay constant, so will the output.

5. Plug in the brushless fan across the Vdd supply and aim it directly toward the canister. The moving air represents a change in the disturbance on your process. According to theory, heat will be removed from the process at a greater rate and the new stable temperature will be lower than 101°. With the fan blowing on the canister, try to find the new "correct" drive for this condition. Record your data in line #6 of the table.

Figure 4.12: Open-Loop Control Table

Line#	Condition	% Drive	Temp
#1	Desired Temperature		101°
#2	Ambient Temperature	0	
#3	50 % Drive Temperature	50%	
#4	Full Drive Temperature	100%	
#5	Appropriate % Drive for 101° Without fan disturbance		101°
#6	Appropriate % Drive for 101° With direct fan disturbance		101°
#7	Appropriate % Drive for 101° With partial fan disturbance		101°

6. Finally, leaving the proper setting established in line #6, change the position of the fan so it is blowing less directly on the canister. This represents a medium disturbance level on the system. Assess the situation and make your best guess as to the proper drive setting required by this new condition. Program the BASIC Stamp for this drive level. Once the system stabilizes, record your results in line #7 of the table.

Challenge!

1. Select a new "desired temperature" for your system. Predict and program an open-loop drive value that will maintain this temperature.

2. Place a couple of glass marbles in your canister. See how increasing the mass of the system affects the response and the drive setting necessary to maintain the new condition. What conclusions can you draw from the system's behavior?

There are many variables that can affect the relationship of drive level and temperature in your small environment. Given some time to experiment and become familiar with the dynamic relationship between temperature, drive level, and disturbances, you could get pretty good at assessing the conditions and setting the right amount of drive in an open-loop manner. As we see, however, if any condition of our process changes, so will the output. Open-loop control can be useful in some applications. When a process requires that its output remain constant for all conditions, then closed-loop control must be employed. In closed-loop control, action is taken based on an evaluation of the measurement and the desired setpoint. This evaluation results in what is called an "error signal."

Questions and Challenge

1. Give two examples of continuous process control other than those given in the text.

2. How is the drive level in open-loop control determined?

3. What is the primary advantage of open-loop control?

4. What is the primary disadvantage of open-loop control?

5. The ADC0831 will convert a range of analog input to one of 256 possible binary values. The number 256 identifies the _____ of the converter.

6. The purpose of the chip select and clock lines to the ADC0831 are to _____ the conversion process.

7. If the LM34 were placed in a 98.6-degree environment, the expected output would be _____volts.

8. In pulse-width modulation, the amount of drive action is based on the _____ of time ON over the total time.

9. If a 40-watt heater were pulse-width modulated at a 75% duty cycle, the average power consumed would be _____ watts.

10. When disturbances change in an open-loop process, so does the _____.